UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

ZIILABS INC., LTD., Plaintiff, v. SAMSUNG ELECTRONICS CO. LTD.; SAMSUNG ELECTRONICS AMERICA, INC.; SAMSUNG TELECOMMUNICATIONS, AMERICA, LLC; SAMSUNG AUSTIN SEMICONDUCTOR, LLC; and APPLE INC. Defendants.	Case No. 2:14-cv-00203-JRG-RSP

DEFENDANTS' SUPPLEMENTAL CLAIM CONSTRUCTION BRIEF

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Page

CASES

I. DISPUTED CLAIM TERMS

A. U.S. Patent No. 7,710,425 - "invisibly to the host processor" (Claims 1, 2) / "invisibly to said CPU" (Claims 6, 11)

Claim Term	Defendants' Proposed Construction	Plaintiff's Proposed Construction
"invisibly to the host processor"	Without awareness of the host processor	Without intervention of the host processor
"invisibly to said CPU"	Without awareness of said CPU	Without intervention of the CPU

The parties dispute whether "invisibly" means without awareness by the host/CPU (as Defendants propose) or merely that there is lack of intervention by the host/CPU (as Plaintiff proposes). The plain meaning, specification, and prosecution history all support Defendants' construction and refute Plaintiff's.

Defendants' construction is fully supported by the specification's explanation that the term "invisibly" means "without awareness":

> In particular, the present application discloses a computer system in which a graphics accelerator unit manages page faulting of texture data ***invisibly to the host processor***.
>
> When a logical page fault occurs and the page of texture is in the second level of memory (i.e. the host's physical memory), it will be fetched in automatically by the graphics memory manager, ***and the host is not aware anything has happened***.

'425 at 6:1-8.[1] During prosecution, the applicants also emphasized this absence of host/CPU "awareness" in distinguishing prior art, arguing that "[t]he ***present inventions*** accomplish [removing the host processor from the 'page fault' routine] by retrieving data from the host's physical memory without any intervention from the host, and ***the host is not aware that anything has happened***." Ex. 23,[2] '425 FH, 4/25/2005 Appeal Br. at 10 (emphasis added).

[1] Emphasis added unless otherwise noted.
[2] All citations to Exhibits 1-45 refer to the exhibits in support of Defendants' Responsive Claim Construction Brief, filed March 6, 2015 (Dkt. No. 151). Citations to Exhibits 46-62 refer to additional exhibits in support of the instant brief.

Although it is also discussed in the intrinsic evidence, lack of intervention by the host is not a construction, but rather a consequence, of the action being "invisible" to the host. Indeed, the sole inventor confirmed that "'invisibly' . . . [i]n this context . . . means that the host doesn't know it's happening." Ex. 46, Baldwin Tr. at 144:18-21.

Plaintiff's construction contradicts the plain meaning of the disputed claim terms, and ignores portions of the intrinsic record. As an initial matter, Plaintiff's construction conflicts with the plain meaning of the term "invisibly" because it reads on a system where managing (or performing) page faulting ***is visible*** to the host processor but the host processor does not intervene. Plaintiff's construction would improperly cover a system where the host does not intervene even though it ***is aware*** something has happened. This is inconsistent with the preferred embodiment of the '425 Patent, in which "the page of texture . . . will be fetched in automatically by the graphics memory manager, and the ***host is not aware anything has happened***." '425 at 6:4-8; *see, e.g.*, *Epos Techs. Ltd. v. Pegasus Techs. Ltd.*, 766 F.3d 1338, 1347 (Fed. Cir. 2014). By contrast, Defendants' construction conforms to the specification's disclosure, the prosecution history, and the inventor's testimony, and should therefore be adopted.

B. U.S. Patent No. 5,831,637

1. "3D graphics request code" (Claim 1)

Defendants' Proposed Construction	Plaintiff's Proposed Construction
3D drawing commands from a host processor	3D graphics commands sent to a graphics engine

Although the parties agree that the claimed "3D graphics request code" includes commands related to drawing 3D graphics, the parties' dispute centers on whether the claimed 3D graphics request code originates from a host processor (as Defendants propose) or whether to ignore the question of where the commands are received from (as Plaintiff proposes).

By requiring an "input" for receiving a "request," the claim requires that an external source originate "3D graphics request code." As known in the art, a request is a "transaction that is generated by a requester, to initiate an action on a responder." Ex. 47, IEEE Dictionary at 911. The identity of the responder is the "graphics engine" as defined by the claim language requiring that "the graphics engine hav[e] a **request code input for receiving 3D graphics the request code**." Defendants' proposed construction, consistent with the patent specification, clarifies that 3D drawing requests originate from a host processor. The specification repeatedly confirms that the external source for these requests is a host processor: "[t]he graphics engine 22 is thus an ASIC **that receives requests from a host processor** via the PCI system bus 201 . . . [r]equests include graphic primitives (points, lines and triangles), rectangular fill, get/put pixel data, blits, and control requests" ('637 at 3:50-56), and "[t]he graphics engine 22 is request-based, in that **it receives requests from a host processor** to perform draw, data movement, and control operations affecting the frame buffer" (*id.* at 6:63-65); *see also id.* at Abstract; 1:19-23; 2:56-3:3; Fig. 2. Similarly, while distinguishing the claimed arrangement over the prior art, the applicants emphasized the communications path of the graphics requests from the host processor via the bus to provide commands from the host:

> Referring to Applicants' FIG. 2, in a preferred embodiment of the present invention, the first stage [is] generation of **a graphics request that is passed over host bus 201 to the graphics processor board**. This request is then placed in a FIFO 2l that accumulates graphics requests, for processing by graphics engine 22.

Ex. 2, '637 FH, 7/16/1997 Amend. at 3. The applicants further explained that because requests come from the host, the invention covers the control of "the progression of requests from receipt over host bus 201, through delivery to the frame buffer" to allow priority processing of video. *Id.* The specification discloses that bus 201, which the applicants identified as the conduit for graphics requests, is the conduit for "requests from a host processor." '637 at 3:51. One of the

inventors of the '637 Patent confirmed that the claimed request code is "received from a host processor." Ex. 48, Young Tr. at 98:4-7. Defendants' construction clarifies, consistent with the intrinsic record, what is meant by the claimed "request"—it is a command from a host processor.

By failing to identify any origin of the 3D graphics request code, Plaintiff's construction effectively vitiates the word "request" and repeats language already present in the claim. Claim 1 reads: "the graphics engine having a request code input for receiving the 3D graphics request code." The claim already identifies the graphics engine as the recipient of the 3D graphics request code. Thus, Plaintiff's construction, which merely repeats that the graphics engine receives the request code rather than saying anything about the request (*i.e.*, where it originates), adds nothing to help the understanding of the term. *See Thomas Swan & Co. Ltd. v. Finisar Corp.*, No. 2:13-cv-00178-JRG, 2014 WL 2885296, *9 (E.D. Tex. June 25, 2014). Defendants' construction is consistent with the intrinsic record and should be adopted.

2. "graphics engine controller" (Claims 1, 2, 5)

Defendants' Proposed Construction	Plaintiff's Proposed Construction
Controller component within the graphics engine	Plain and ordinary meaning. In the alternative "logic that directs or regulates the graphics engine."

The primary dispute is whether the graphics engine controller is a component of the graphics engine (as Defendants propose), or if it can encompass any undefined logic (presumably software or hardware) that, at some point, provides any direction or regulation to the graphics engine (as Plaintiff proposes).

The intrinsic record confirms that the "graphics engine controller" is part of the graphics engine. As an initial matter, it is a matter of plain English and common sense that the "graphics engine controller" would actually be a part of the graphics engine. Claim 1 requires that the "graphics engine controller" is responsible for interrupting processing of 3D graphics request

code: the "graphics engine controller . . . interrupt[s] processing by the graphics engine" and the "graphics engine controller permit[s] priority processing of the digital video data." '637 at Claim 1. *See Phillips v. AWH Corp. Inc.*, 415 F.3d 1303, 1314 (Fed. Cir. 2005). According to the specification, the **graphics engine** ("GE") performs this claimed interrupting, requiring that this functionality exists within the graphics engine: "[w]hen data is available to be read from the Video FIFO, **the GE looks for an interruptible point** at which to stall the normal request stream." '637 at 19:30-35. Because the **graphics engine** finds interruptible points, and the "graphics engine controller" performs this function in the claim, it follows that the graphics engine controller must be located within the graphics engine. The inventors confirm that "[t]he multiplexer is what looks for the interruptible point," "the multiplexer is within the graphics engine," and "the graphics engine would have control circuitry for handling the interrupting mechanism provided by the multiplexer." Ex. 49, Deming Tr. at 66:16-68:12.[3]

Plaintiff's construction fails because it contradicts the plain meaning and applicants' statements to the PTO. During prosecution, the examiner determined that interrupting a graphics engine would have been obvious in view of the Guttag reference's teaching "that the **host interface controls the timing** of data transfers . . . and that a request code may be utilized to direct the graphics engine to interrupt its processing." Ex. 2, '637 FH, 7/16/97 Amend. at 3. In response, the applicants argued that in the prior art, "[n]o provision is made **for the graphics engine** to look for an interruptible point . . . so as to prevent stalling." *Id.* at 4. And specifically, the applicants argued that Guttag did not teach "how to modify a graphics processor board" to support "a control arrangement for allowing such preemptive processing of incoming video data." *Id.* at 5. Plaintiff's construction should be rejected because it would allow the controller

[3] *See also* Ex. 48, Young Tr. at 99:23-100:4, stating that the graphics engine controller is the "same thing" as the graphics engine.

logic that directs the graphics engine to come from anywhere, including the host, notwithstanding the applicants' statements to the contrary. *Id.* at 5.

3. "span break" (Claim 5)

Defendants' Proposed Construction	Plaintiff's Proposed Construction
End of a horizontal sequence of adjacent pixels in a graphics primitive	an interruptible point in the request stream

The parties dispute whether a span break refers to the end of a **horizontal** sequence of adjacent pixels **in a graphics primitive** (as Defendants propose) or whether the term should be rewritten to cover **any** point where graphics processing is stopped (as Plaintiff proposes). Defendants' construction reflects the intrinsic record, while Plaintiff's construction both reads out the word "span"—a well-understood term of art in 3D graphics processing—from the claim and improperly broadens the claim to cover arbitrary interrupting of graphics processing.

Consistent with Defendants' construction, the specification confirms that a "span" means a sequence of horizontally adjacent pixels, and that these pixels are part of a graphics primitive. As the specification describes, "requests are broken down into **'span' requests**—requests to read or write **a horizontal sequence of adjacent pixels**." '637 at 3:57-59. The specification also unambiguously shows that the sequence of adjacent pixels in a span is "in a graphics primitive." The specification states "The span generation block performs the span calculation part of the rendering process **for triangles and lines**" (*id.* at 3:31-33) and identifies triangles and lines as types of graphics primitive (*id.* at 3:51-56; *see also id.* at 6:29-34). In fact, the purpose of a span is widely understood in the art to define the pixels of a scan line that fall within a primitive, and it would defy this purpose if a span was construed as not being "in a graphics primitive." Foley and van Dam (cited by both Plaintiff and Defendants) defines a span as "pixels on a scan line,"

with endpoints determined by the edges of a graphics primitive, as further illustrated in the accompanying figure:

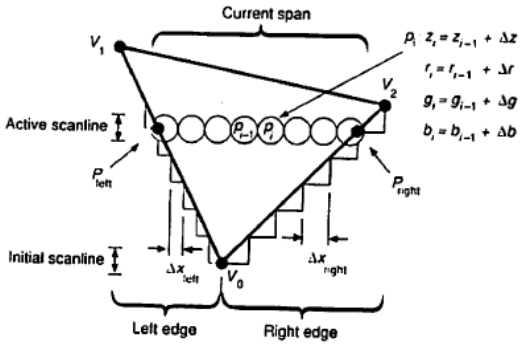

Fig. 18.13 Rasterizing a triangle. Each vertex (V_0, V_1, and V_2), span endpoint (P_{left} and P_{right}), and pixel (p_0, p_1, etc.) has *z*, *R*, *G*, and *B* components.

Ex. 50, Computer Graphics: Principles and Practice (1990), 883-84; *see also id.* at 92-95, 882.[4]

Moreover, the intrinsic evidence supports Defendants' construction that a "break" is the "end of a" sequence of adjacent pixels. Claim 5 recites that the graphics engine interrupts processing "at a span break." The specification explains that "the [graphics engine] 42 has the capability of breaking spans into subspans"—shorter sequences of adjacent pixels—so that the graphics engine can interrupt normal graphics processing in favor of video. '637 at 19:21-35. When the graphics engine locates a "span break," also called a "span boundary," it can interrupt processing. *Id.* at 19:44-46.

The prosecution history makes clear that the applicants intended "break" to mean the end of a particular pixel sequence. To overcome the Guttag reference, the applicants explained that the graphics engine must look for the end of a sequence to prioritize video data processing: "No provision is made [in the prior art] for the graphics engine to look for an interruptible point (i.e. to break up large requests) so as to prevent stalling." Ex. 2, '637 FH, 7/16/97 Amend. at 4. The

4 Inventor testimony confirms the definition of a "span" in Foley and van Dam is consistent with the specification's use of the word "span" at column 3, lines 57 through 59. Ex. 49, Deming Tr. at 103:15-105:23.

applicants repeatedly argued that the controller broke up requests into smaller requests, indicating that "break" refers to the end of a sequence. *See* Ex. 2, '637 FH, 7/16/97 Amend. at 4 ("**By breaking up the graphics requests**, the graphics engine will not stall during processing of a graphics request."); Ex. 51, '637 FH, 3/26/98 Amend. at 4 ("the controller ensures against pipeline stalling by **breaking graphics requests into smaller requests** that will fit in the pipeline"); Ex. 52, '637 FH, 4/28/98 Amend. at 4. The graphic engine uses a "span boundary" (the end of a span) to signal that processing a graphics primitive has reached an interruptible point where a request has been broken into a smaller request. *See id.*; *see also* '637 at 19:44-45.

By rewriting the claim term to cover ***any*** interruptible point where graphics processing is stopped, Plaintiff's construction effectively reads the term "span" out of the claim and contradicts the intrinsic record. As described above, the specification discloses breaking spans into subspans and further describes that graphics processing is interrupted at these span breaks. '637 at 19:44-45. In doing so, the specification describes how the invention uses a span break, but it is not defining ***any*** point at which graphics processing is interrupted as a span break. To the contrary, during prosecution, the applicants noted that if graphics processing was "**arbitrarily interrupted** in favor of video data," then "video data commonly would be lost," Ex. 51, '637 FH, 3/26/98 Amend. at 4. Under Plaintiff's construction, however, **any** arbitrary point where graphics processing is interrupted qualifies as a "span break." The intrinsic record is clear—a span break is not a theoretical "interruptible point" without discrete contours; it is a particular one, which comes at the end of a sequence of adjacent pixels. Plaintiff's construction thus ignores and indeed contradicts the intrinsic record's explicit teaching that to properly process graphics and video data, graphics processing is specifically interrupted at breaks in spans rather than at any arbitrary point.

C. U.S. Patent No. 7,187,383

1. "graphics computational units" ('383 Claims 1, 10)

Defendants' Proposed Construction	Plaintiff's Proposed Construction
Computational subunits within a single graphics processor	Computational units that form part of a graphics processor

The parties agree that this term requires a graphics processor made up of computational units. But whereas the intrinsic evidence demonstrates that the claimed invention requires a **single** graphics processor made up of multiple computational subunits (as in Defendants' construction), Plaintiff's proposal would blur the distinction between graphics processors and their computational units, which was the basis for distinguishing prior art during prosecution.

The '383 Patent itself, from the claim language that uses the singular "graphics processor" to the specification and figures that all describe a single graphics processor with multiple computational units, supports Defendants' construction. *See, e.g.*, '383 at Fig. 1; 2:25-35; 2:55-58; 3:31-41. During prosecution, the applicants also explained that the invention relates to a single processor as opposed to multiple processors. In their appeal brief from a final rejection of claims, the applicants explained that "Claim 1 claims in part 'a plurality of paralleled graphics computational units' which is part of a single processor" and that all claims "are directed towards . . . 'a single processor, with parallel functions.'" Ex. 53, '383 FH, 3/4/05, Appeal Br. at 11. Applicants also pointed to Figure 1 as illustrating the "single processor" of the claimed invention. *Id*. at 11-12. Against this backdrop, applicants then distinguished the Brent reference, stating: "the prior art teaches away from the present inventions **as it . . . requires a plurality of processors** in order to function. The Brent [prior art] patent requires a multiple of separate, fully functional processors in order to function." *Id*. at 13. The applicants concluded emphatically that "[t]his application deals with the ability to bypass a defective pipeline within a SINGLE processor ('A GRAPHICS PROCESSOR') as opposed to the MULTIPLE ('PLURAL

QUEUE PROCESSORS') processors alluded to in the prior art." *Id*. at 21 (emphasis and all caps original).[5] The intrinsic evidence thus unambiguously dictates a construction requiring multiple computational subunits within a **single** graphics processor.

Defendants also include "subunits" in their construction to clarify that each "computational unit" cannot itself be a standalone unit like a processor, but instead only exists as a subunit within a single graphics processor. This is supported by the claim language and specification, which describe subunits (*e.g.*, texture pipes, vertex processors, etc.) as components within a single graphics processor. *See, e.g.*, '383 at Claim 1, Fig. 1, 2:25-35, 2:55-58, 3:31-41. This is also supported by the applicants' statements during prosecution. In distinguishing Brent, the applicants argued that "processor" and "unit" are "not interchangeable." Ex. 54, '383 FH, 10/5/04 Amend. at 12. Claim 1 also originally recited "subunits" rather than "computational units." Ex. 55, '383 FH, 3/1/02, Original App. at 47. In amending the claim to change "subunits" to "computational units," the applicants explained the amendments as "purely formal amendments, and are believed not to change the scope of these claims." Ex. 56, '383 FH, 5/4/04 Amend. at 16 (emphasis original). Indeed, even after making this amendment, the applicants still referred to the computational units as "subunits." *See, e.g.*, Ex. 53, '383 FH, 3/4/05, Appeal Br. at 20 ("in the present inventions the processor will continue to function . . . by disabling a **subunit**"); *see also id*. at 23. Accordingly, the term "subunit" in Defendants' construction is fully supported by the intrinsic evidence and appropriately clarifies the distinction between the single graphics processor and the computational units within it.

[5] The prosecution history supports the plain meaning construction of the patent, requiring a single processor. But even if ZiiLabs argues and the Court finds that the patent would dictate a different meaning, the statements made during prosecution distinguishing multiple processors in the prior art, and emphasizing that the invention requires a "single processor," qualify as a clear and ambiguous disclaimer of any broader scope. *See, e.g.*, *Vita-Mix Corp. v. Basic Holding, Inc.*, 581 F.3d 1317, 1324 (Fed. Cir. 2009).

2. "task allocation units programmed to bypass defective ones" ('383 Claim 1)

Defendants' Proposed Construction	Plaintiff's Proposed Construction
Units running software to recognize and avoid allocating tasks to defective graphics computational units	Functional units programmed to distribute tasks only to operative graphics computational units

The parties agree that this disputed term refers to "units" that allocate or distribute tasks to certain computational units. But whereas Defendants' construction simply adopts the ordinary meaning of the claim, Plaintiff's proposal omits the critical aspect of the claim term of bypassing "defective" units and, in its place, adds unnecessary complexity and confusion to the term.

Claim 1 recites that the "task allocation units" are "programmed to bypass" defective units. The specification explains "programmed to bypass" in this context means running software that recognizes defective units and avoids allocating tasks to those units. For example, the specification states that the invention "use[s] **software** device drivers which **recognize** which of the parallel units are functional, and use only the functional paths." '383 at 3:48-50. The specification further provides an example where the computational units are "texture pipes" and software is used to recognize which are defective and to be avoided: "the number of texture pipes is **transparent to the software** and the Texture Switch Unit can **avoid** using texture pipes with manufacturing defects." *Id.* at 14:33-40. Thus, Defendants' construction provides "programmed to bypass" with its ordinary meaning in the context of the '383 Patent. *See World Class Tech. Corp. v. Ormco Corp.*, 769 F.3d 1120, 1123 (Fed. Cir. 2014) ("We generally give words of a claim their ordinary meaning in the context of the claim and the whole patent document").

Plaintiff's proposal attempts to rewrite the claim term and render the surrounding claim language superfluous. Plaintiff changes the claim language from reciting which units are bypassed or avoided (*i.e.*, the "defective" units) to now reciting the opposite, namely, which

units are affirmatively used (*i.e.*, "operative" units). Plaintiff's proposal is also incorrect because it implies that tasks cannot be distributed to merely inoperative computational units. But units may be inoperative for many reasons—such as being intentionally turned off—even if they have no manufacturing defects and thus are not "defective." Plaintiff's proposal is therefore contrary to the specification that repeatedly describes the claimed invention using the specific terms of avoiding or overcoming "defective" units or units with manufacturing "defects" or faults. '383 at 2:42-44 (describing an advantage of the invention as "ability to use partially defective die as fully functioning parts with lower performance"); 14:34-40 (describing ability to "avoid using texture pipes with manufacturing defects" to "improve the effective manufacturing yield"); *see also id.* at 2:33. Moreover, Plaintiff improperly construes the disputed term to repeat language already present in the very next limitation in the claim, which recites "distribute incoming tasks only among operative ones of said units." *Id.* at Claim 1. Its proposal should therefore be rejected because constructions that renders claim terms superfluous are disfavored. *See Douglas Dynamics, LLC v. Buyers Products Co.*, 717 F.3d 1336, 1350 (Fed. Cir. 2013).

D. U.S. Patent No. 5,835,096 - "character data" (Claim 5)

Defendants' Proposed Construction	Plaintiff's Proposed Construction
Data representing text characters	Image data representing a graphic symbol (as a glyph or alphabet letter)

The parties dispute whether "character data" is data specifically representing "text characters" (as Defendants propose) or whether the term should be re-written to include any "image" that happens to contain graphic symbols (as Plaintiff proposes). The claims, specification, and inventor testimony all confirm that Defendants' construction is correct and Plaintiff's proposed construction improperly redefines the claim.

Consistent with how the '096 Patent uses the term, Defendants' construe "character data" as a specific type and format of data that yields text:

> **A character can be a one bit texture map** and the foreground and background colours set up in the TexelLUT table in the Texture Read Unit. If the background colour has a different alpha value than the foreground colour then the alpha test unit can be used to terminate the background fragments **giving transparent text**. A deeper texture map can be used **so that text can have more foreground colours or be antialiased** (by using different alpha values or edge colours).

'096 at 61:28-35. The specification clarifies that data representing characters is treated differently than other data; it is a specific type of data. Defendants' construction is also consistent with the understanding of persons skilled in the art, as reflected by technical dictionaries and the named inventor. *See, e.g.*, Ex. 57, New World Dictionary of Computer Terms 83 (1994) (defining "character data" as "[d]ata in the form of letters and special characters."). The '096 Patent's inventor confirmed that "you have to be able to identify the -- the data that represents a particular text character in order for it to qualify as character data." Ex. 46, Baldwin Tr. 264:15-21.

Plaintiff improperly attempts to re-write the claim term by transforming "character data" to any "image data" that happens to have graphic symbols in it. However, the '096 Patent's claims carefully distinguish between different data types: "character data" (claim 2) "tile pattern data" (claim 3), "texture data" (claim 5), "pattern data" (claim 6), and "stipple pattern data" (claim 7). From this, it is evident that the patentee intended to differentiate between these data types and it is thus improper to replace one type of data, "character data," with another, "image data." The specification discusses "image data" only twice, and neither time is related to character data. '096 at 24:8-11; 54:14-16. The specification also describes character data as text characters. *Id.* at 61:28-35 ("A character can be a one bit texture map . . . [or a] deeper texture

map can be used so that text can have more foreground colours."). The inventor explained why image data is different than character data:

> Q: And that's because the data doesn't represent a particular text character?
>
> A: It's because it's just an image.
>
> Q: So an image that might have alphabet letters within it isn't necessarily character data?
>
> A: That's correct. . .
>
> * * * *
>
> Q. . . . [J]ust because an image might contain alphabet letters or icons, that's not necessarily character data within the context of the '096 Patent?
>
> A. ***That's true*** if you cannot readily access those individual characters to extract them and use them in a different image.

Ex. 46, Baldwin Tr. 263:15-264:13. In sum, one of ordinary skill would understand that "character data" is a type of data representing text characters (as opposed to an image containing some text). Nothing in the intrinsic record justifies rewriting "character data" to encompass image data as Plaintiff proposes.

E. U.S. Patent No. 8,144,156 - "decouples operations" ('156 Claim 6)

Defendants' Proposed Construction	Plaintiff's Proposed Construction
Allows operations not to run at a fixed delay	Plain and ordinary meaning Alternatively, "allows asynchronous operations"

The parties appear to agree that "decouples operations" should be construed similarly to "logically asynchronous to," which the parties have briefed separately.[6] Accordingly, Defendants refer the Court to the briefing for "logically asynchronous to" for an explanation of why their construction for "decouples operations" is correct. *See* Def. Br. 26-28. Briefly

[6] Defendants' construction for "decouples operations" is similar to their construction for "logically asynchronous to," with both permitting non-"fixed delay" operations. *See* Def. Br. 26. Plaintiff argues the two terms refer to the same concept (Pl. Br. 24-25), and its alternative proposal for "decouples operations" is "allows asynchronous operations."

summarizing those reasons here: (1) the specification distinguishes the "asynchronous" or "decoupled" relationship of the sequencer and PEs in the claimed invention from the fixed delay or "lock step" relationship in the prior art (*e.g.*, '156 at 3:60-4:1); (2) applicants during prosecution likewise distinguished "fixed delay" in prior art from the claimed "asynchronous" or "decoupled" limitations (Ex. 40, '156 FH, 1/31/08 Amend. at 14-15); and (3) the named inventor equated "asynchronous" with "decoupled," and confirmed that "asynchronous" means not at a "fixed delay" (Ex. 46, Baldwin Tr. at 180:10-12, 200:2-16, 204:14-16). Plaintiff's proposals of no construction or alternatively replacing "decouples" with "asynchronous" should be rejected because the claim would have unclear meaning to a jury in the context of the '156 Patent without construction. *See Abbott Labs. v. Sandoz, Inc.*, 544 F.3d 1341, 1360 (Fed. Cir. 2008) ("Claim construction is for the purpose of explaining and defining terms . . . claims are construed as an aid to the decision-maker").

II. CONCLUSION

Defendants respectfully request that the Court adopt Defendants' constructions.

Dated: March 11, 2015

Respectfully submitted,

/s/Nicholas J. Whilt

Michael Jones (Texas Bar No. 10929400)
Lead Attorney
mikejones@potterminton.com
Allen Gardner (Texas Bar No. 24043679)
allengardner@potterminton.com
Potter Minton
110 N. College
Tyler, Texas 75702
Telephone: (903) 597-8311
Fax: (903) 593-0846

George A. Riley (Cal. Bar No. 118304)
griley@omm.com
Mark Liang (Cal. Bar No. 278487)
mliang@omm.com
O'MELVENY & MYERS LLP
Two Embarcadero Center, 28th Floor
San Francisco, CA 94111
Telephone: 415-984-8700
Facsimile: 415-984-8701

Brian M. Berliner (Cal. Bar No. 156732)
bberliner@omm.com
Ryan K. Yagura (Tex. Bar No. 24075933)
ryagura@omm.com
Nicholas J. Whilt (Cal. Bar No. 247738)
nwhilt@omm.com
John K. Woo (Cal. Bar No. 281132)
jwoo@omm.com
O'MELVENY & MYERS LLP
400 S. Hope Street
Los Angeles, CA 90071
Telephone: 213-430-6000
Facsimile: 213-430-6407

Cameron Westin (Cal. Bar No. 290999)
cwestin@omm.com
Sarah A. Pfeiffer (Cal. Bar No. 278205)
spfeiffer@omm.com
O'MELVENY & MYERS LLP
610 Newport Center Drive, 17th Floor
Newport Beach, CA 92660

Telephone: 949-823-6900
Facsimile: 949-823-6994

Attorneys for Defendants Samsung Electronics Co., Ltd., Samsung Electronics America, Inc., Samsung Telecommunications America, LLC, and Samsung Austin Semiconductor, LLC

Dated: March 11, 2015

Respectfully submitted,

/s/Marcus E. Sernel, P.C.

Melissa R. Smith
State Bar No. 24001351
GILLAM & SMITH LLP
303 S. Washington Avenue
Marshall, Texas 75670
Telephone: (903) 934-8450
Facsimile: (903) 934-9257
E-mail: Melissa@gillamsmithlaw.com

Gregory S. Arovas, P.C. (*admitted pro hac vice*)
Todd M. Friedman, P.C. (*admitted pro hac vice*)
KIRKLAND & ELLIS LLP
601 Lexington Avenue
New York, New York 10022
Telephone: (212) 446-4800
Facsimile: (212) 446-4900
E-mail: greg.arovas@kirkland.com
E-mail: todd.friedman@kirkland.com

Marcus E. Sernel, P.C. (*admitted pro hac vice*)
Colleen M. Garlington (*admitted pro hac vice*)
KIRKLAND & ELLIS LLP
300 N. LaSalle Street
Chicago, Illinois 60654
Telephone: (312) 862-2000
Facsimile: (312) 862-2200
E-mail: marc.sernel@kirkland.com
E-mail: colleen.garlington@kirkland.com

Attorneys for Defendant Apple Inc.

CERTIFICATE OF SERVICE

The undersigned hereby certifies that all counsel of record who are deemed to have consented to electronic service are being served with a copy of this document by the Court's CM/ECF system per Local Rule CV-5(a)(3) on March 11, 2015.

/s/Marcus E. Sernel, P.C.